Reinforcement Learning in Categorical Cybernetics

Jules Hedges, Riu Rodríguez Sakamoto

Mathematically Structured Programming group,
Department of Computer and Information Sciences,
University of Strathclyde

riu.rodriguez-sakamoto@strath.ac.uk

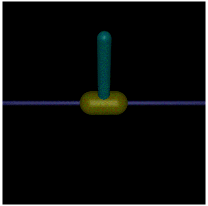
7th International Conference on Applied Category Theory,
Oxford

June, 2024

Introduction

1. Search for optimal control solutions is difficult:

- LQR: Linear system dynamics, quadratic cost.
  Analytic closed solution
- MDP and nonlinear dynamics, arbitrary cost.
  Iterative solution
- RL: Unknown environment dynamics, unknown cost.
  What is the structure for solution methods here?



[MuJoCo]

Introduction

Design of RL algorithms is a craft. There is a discrepancy in specificity between pseudocode and (informal) diagrams. Can we do better?

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

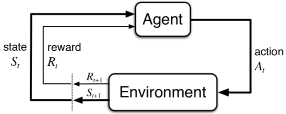


Figure 3.1: The agent–environment interaction in a Markov decision process.

Algorithm 1 PPO, Actor-Critic Style

for iteration$=1,2,\ldots$ **do**
  for actor$=1,2,\ldots,N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1,\ldots,\hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \le NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$
where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

Control problems

$$\text{Environment: } \begin{cases} \text{state space} & S & \text{transition} & t : S \times A \to S \\ \text{action space} & A & \text{immediate reward} & r : S \times A \to \mathbb{R} \end{cases}$$

Objective: Choose a policy $\pi : S \to A$ to optimize long-run reward.
For a fixed $0 < \gamma < 1$, the long-run reward is given by a discounted sum

$$V_\pi : S \to \mathbb{R} \qquad V_\pi(s_0) = \mathbb{E}_{s' \sim t(s,\pi(s))} \sum_{k=0}^{\infty} \gamma^k r(s_k, \pi(s_k))$$

$$Q_\pi : S \times A \to \mathbb{R} \qquad Q_\pi(s_0, a_0) = \mathbb{E}_{s' \sim t(s,\pi(s))} \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k)$$

Dynamic Programming (DP)	Known environment	Backward induction
Reinforcement Learning (RL)	Unknown environment	?

Control problems

Examples:

- DP: Value improvement (MDP known)

$$V(s) \leftarrow \mathbb{E}_{(r,s')\sim t(s,\pi(s))}[r + \gamma V(s')]$$

$$V(s) \leftarrow \underbrace{\mathbb{E}_{(r,s')\sim t(s,\pi(s))}[r + \gamma V(s')]}_{\text{update target}} \qquad (\alpha = 1)$$

- RL: The SARSA algorithm (MDP unknown, aka “model-free”): Sampling (s, a, r, s', a')

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[\underbrace{r + \gamma Q(s',a')}_{\text{update target}}]$$

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[\underbrace{r + \gamma Q(s',a')}_{\text{update target}}]$$

Separation of concerns: Update target computation and update operation.

Optics 101

Lenses and optics a way to specify bidirectional processes as morphisms.[1]

$$\mathbf{Lens}(\mathcal{C})\left(\begin{pmatrix}X\\X'\end{pmatrix},\begin{pmatrix}Y\\Y'\end{pmatrix}\right)=\mathcal{C}(X,Y)\times\mathcal{C}(X\times Y',X')$$

$$\mathbf{Optic}(\mathcal{C})\left(\begin{pmatrix}X\\X'\end{pmatrix},\begin{pmatrix}Y\\Y'\end{pmatrix}\right)=\int^{M:\mathcal{C}}\mathcal{C}(X,Y\otimes M)\times\mathcal{C}(M'\otimes Y',X')$$

States and continuations: When the monoidal unit of $\mathcal{C}$ is terminal (e.g. Markov categories),

$$\mathbf{Optic}\left(I,\begin{pmatrix}X\\X'\end{pmatrix}\right)\cong\mathcal{C}(I,X)\qquad\mathbb{K}\begin{pmatrix}X\\X'\end{pmatrix}=\mathbf{Optic}\left(\begin{pmatrix}X\\X'\end{pmatrix},I\right)\cong\mathcal{C}(X,X')$$

Let $\mathbb{K}:\mathbf{Optic}^{\mathrm{op}}\to\mathbf{Set}$ be the continuation functor, represented by I.

[1] Here the forwards maps are above (sorry!)

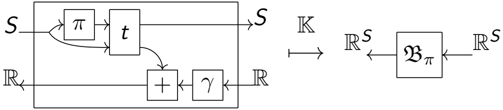
Bellman operators

$$\mathbf{Optic}\left(\begin{pmatrix}X\\X'\end{pmatrix},\begin{pmatrix}Y\\Y'\end{pmatrix}\right)=\int^{M:\mathcal{C}}\mathcal{C}(X,Y\otimes M)\times\mathcal{C}(M'\otimes Y',X')$$

The value improv. map is the evaluation of the Bellman operator on the state s.

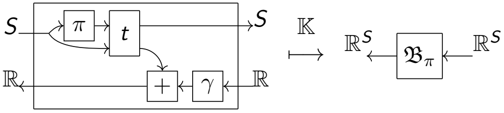
$$\mathfrak{B}_\pi : L^\infty(S)\to L^\infty(S) \qquad \text{in } \mathbf{CMet}$$
$$\mathfrak{B}_\pi : \mathbb{R}^S\to\mathbb{R}^S \qquad \text{in } \mathbf{Set}$$
$$\mathfrak{B}_\pi(V)(s)=\mathbb{E}_{(r,s')\sim t(s,\pi(s))}[r+\gamma V(s')]$$

A (linear) Bellman operator $\mathfrak{B}_\pi$ is the $\mathbb{K}$-image of the Bellman operator optic.

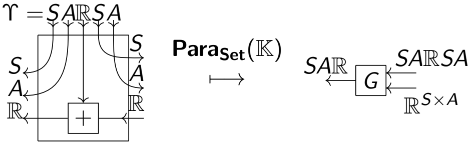


Is there a similar construction for RL algorithms?

Bellman operators



Bellman operator (Value Iteration)



parametric Bellman function (SARSA)

Para$_{\mathbf{Set}}(\mathbb{K})$ is the (external) **Para** lifting of $\mathbb{K}$.
A (linear) parametric Bellman function G is **Para**$_{\mathbf{Set}}(\mathbb{K})$-image of a parametrised optic.

$$\mathbf{Para}_{\mathbf{Set}}(\mathbb{K})(\mathfrak{B}) : \Upsilon \times \mathbb{R}^{S\times A} \to S \times A \times \mathbb{R} \qquad \hookrightarrow \mathbb{R}^{S\times A} \qquad \text{in } \mathbf{Set}$$

$$((s,a,r,s',a'),Q) \mapsto (s,a,r+\gamma Q(s',a')) \qquad \mapsto I_{(s,a)}[r+\gamma Q(s',a')]$$

Policies

Recall the SARSA algorithm (MDP unknown):

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma Q(s', a')]$$

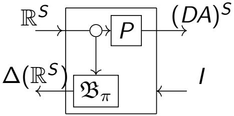
How do we get the reward r', next state s', next action a' if the MDP is unknown?

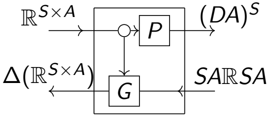
$$P : \mathbb{R}^{S\times A} \to (DA)^S$$
$$Q \mapsto \pi(s) = \mathrm{argmax}_a\, Q(s,a)$$

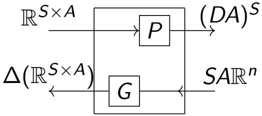
When $S = 1$, $P : (S \to A \to \mathbb{R}) \to (S \to DA) \;\cong\; (A \to \mathbb{R}) \to DA$.

Models

Learning structure of models:







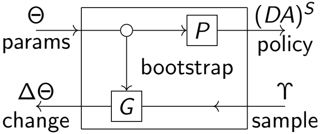
Bootstrap
(Dynamic Programming)

Bootstrap + sample
(Temporal Difference, like SARSA or Q-learning)
$S \times A \times \mathbb{R} \hookrightarrow \Delta(\mathbb{R}^{S\times A})$

Sample
(Monte Carlo)

Reinforcement Learning model lens

model update



interaction with environment via an agent

Iteration contexts

$$\mathbf{Optic}(\mathcal{C}) \to \mathbf{Set} \quad \cong \quad \begin{cases} W : \mathcal{C} \times \mathcal{C}^{\mathrm{op}} \to \mathbf{Set} \\ W(M \otimes X, M \otimes Y) \to W(X, Y) \quad \text{nat. in } X, Y \end{cases}$$

For a symm. mon. cat. $\mathcal{C}$, an iteration context for $\mathcal{C}$ is $\mathbb{I} : \mathbf{Optic}(\mathcal{C}) \to \mathbf{Set}$,

$$\mathbb{I}\binom{X}{X'} = \int^{M:\mathcal{C}} \mathcal{C}(I, M \otimes X) \times \mathcal{C}(M \otimes X', M \otimes X)$$

For a representative element $(M, x_0, i) \in \mathbb{I}\binom{X}{X'}$:

- M: state space
- $x_0 : I \to M \otimes X$: initial state
- $i : M \otimes X' \to M \otimes X$: iterator

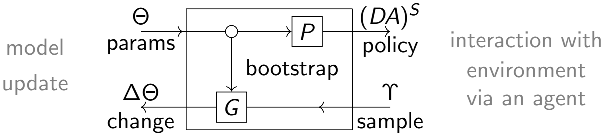
Weighted Para

Let $\mathcal{D}$ be a symm. mon. cat.,
$W : \mathcal{D} \to \textbf{Set}$ a symm. lax mon. functor.
The action of $\int W$ on $\mathcal{D}$ given by $(M, w) \bullet X = M \otimes X$ generates

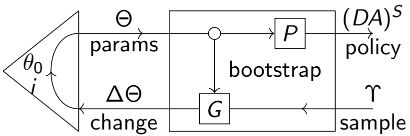
- the bicategory $\textbf{Para}^W(\mathcal{D})$ (morphism: $M \otimes X \to Y$)
- the 1-category $\pi_0^*(\textbf{Para}^W(\mathcal{D}))$ by quotienting invertible 2-cells
  - UP: Freely extends $\mathcal{D}$ with states $\forall X \in \mathcal{D}, \forall w \in F(X).w : I \to X$.

For $W = \mathbb{I} : \textbf{Optic}(\mathcal{C}) \to \textbf{Set}$, the symm. mon. cat. $\textbf{Optic}^{\mathbb{I}}(\mathcal{C}) = \pi_0^*(\textbf{Para}^{\mathbb{I}}(\textbf{Optic}(\mathcal{C})))$ extends $\textbf{Optic}(\mathcal{C})$ with states $I \to \binom{X}{X'}$ defined by elements of $\mathbb{I}\binom{X}{X'}$.

Reinforcement Learning model lens



Reinforcement Learning model lens



interaction with environment via an agent

- Step-size update: $Q_{\text{new}} \leftarrow (1-\alpha)Q_{\text{old}} + \alpha(\text{target})$
- Gradient descent, momentum [2]

[2] Cruttwell, Gavranović, Ghani, Wilson, Zanasi: Categorical Foundations of Gradient-Based Learning (Proc.ESOP 2022)

Model, Agent & Environment

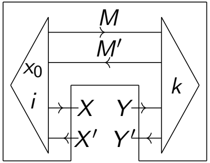
- Model: Optic that parametrises an agent, extended with an update iteration
- Agent: (Model-)parametrised optic
- Environment: Iteration context for optics

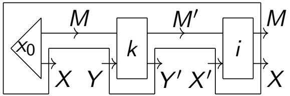
DP: The data defining the environment is the same as the data defining the model.
RL: The model is an approximation of the environment.

Environments: Iteration contexts for optics

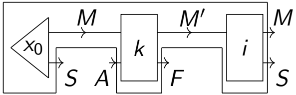
The representable functor $\mathbb{I}_{\mathrm{env}} : \mathbf{Optic}(\mathbf{Optic}^{\mathbb{I}}(\mathcal{C})) \to \mathbf{Set}$ maps an object (X, X', Y, Y') to a set with elements

$$(x_0, k, i) \in \mathbb{I}_{\mathrm{env}}\left(\begin{pmatrix} X \\ X' \end{pmatrix}, \begin{pmatrix} Y \\ Y' \end{pmatrix}\right) \cong \int^{M,M':\mathcal{C}} \mathcal{C}(I, M \otimes X) \times \mathcal{C}(M \otimes Y, M' \otimes Y') \times \mathcal{C}(M' \otimes X', M \otimes X)$$

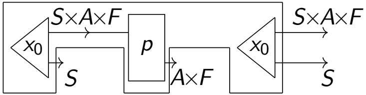




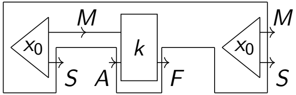
Environments: Examples



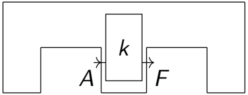
Online (MDP: $M = S$, POMDP)



Offline (dataset, ER)

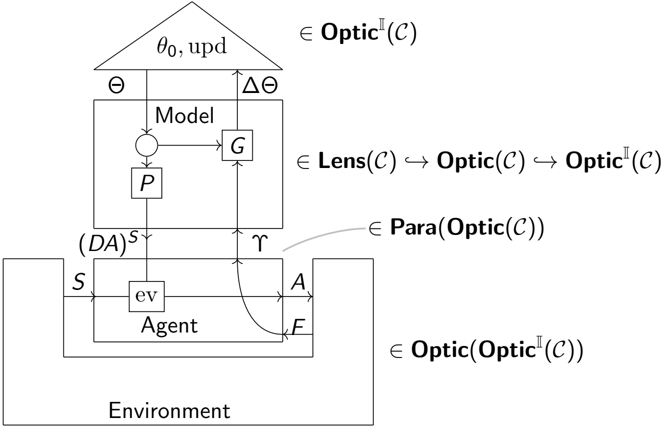


Contextual bandit

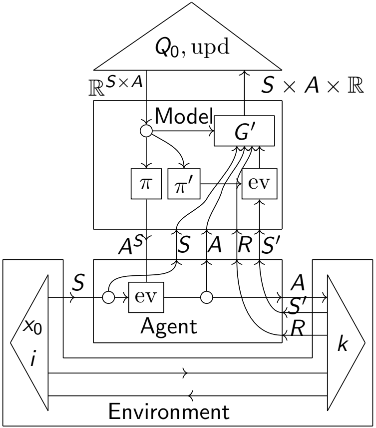


Multi-armed bandit

String diagram: Putting the pieces together



String diagram: Putting the pieces together



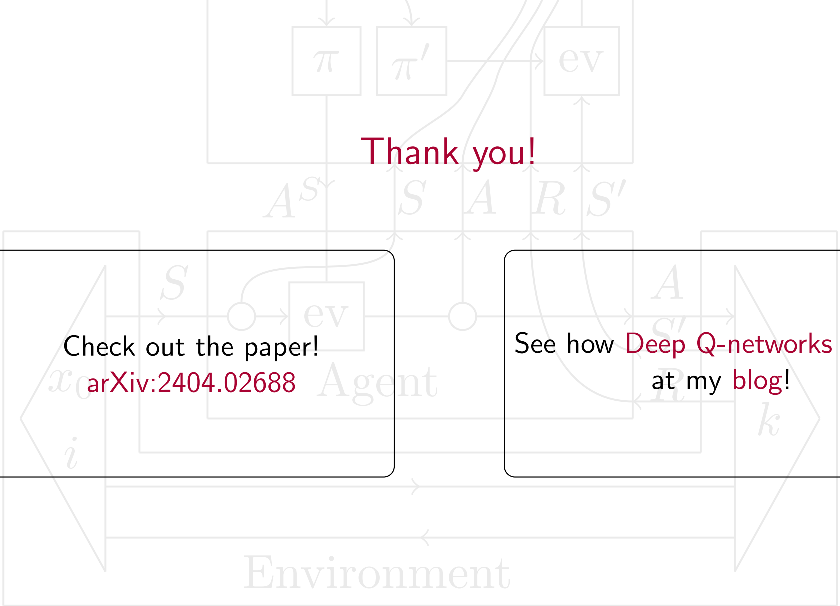
Summary

- Identified main building blocks of RL
- String diagrammatic syntax helps visualise design distinctions
  - Target computation, update operation
  - Linear/non-linear, parametric/non-parametric Bellman operators
  - Learning: Bootstrap vs sampling

Dynamic Programming	Known environment	Backward induction
Reinforcement Learning	Unknown environment	RL lens

  - Online vs offline environments
  - ...
- Work in progress:
  - Compositional solution concepts. Relation to an open systems theory
  - Multi-agent RL
  - Convergence of Q-learning

Thank you!



Check out the paper!
arXiv:2404.02688

See how Deep Q-networks fit here at my blog!

On-policy, off-policy algorithms

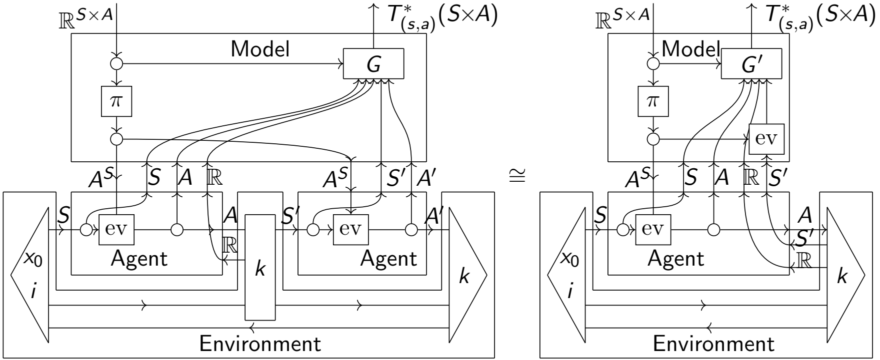


Figure 1: SARSA, an on-policy algorithm.

On-policy, off-policy algorithms

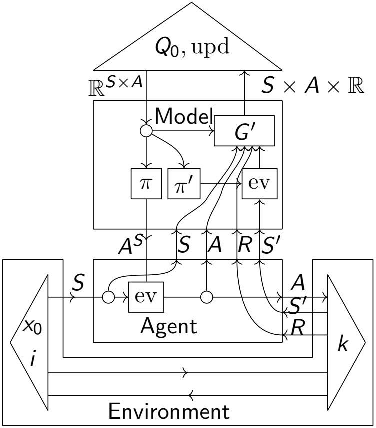


Figure 2: Q-learning, an off-policy algorithm.

Feedback type for learning models

- Dynamic Programming: $\Upsilon = I$

$$V(s) \leftarrow \max_a \mathbb{E}_{(s',r)\sim t(s,a)}[r + \gamma V(s')]$$

- Monte Carlo: $\Upsilon = SA\mathbb{R}^n$ (n-step episodes)

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \sum_{t=0}^{n} \gamma^t r_t$$

- Temporal Difference: $\Upsilon = SA\mathbb{R}SA$

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma Q(s',a')]$$

Ablation maps

Removing either the bootstrapping or the sampling component produces other existing algorithms:

Update	w/o bootstrap	w/o sampling
Q-learning	1-step Monte Carlo	Value iteration
Exp-SARSA	1-step Monte Carlo	Value improvement